Package 'replacer'

July 23, 2025

Title A Value Replacement Utility

Version 1.0.2

Date 2022-08-19

Description Updates values within csv format data files using a custom, User-built csv format lookup file. Based on 'data.table' package.

License GPL-3

Imports data.table(>= 1.14.0)

Depends R(>= 4.1.0)

Suggests knitr, kableExtra, rmarkdown, carData, testthat (>= 3.0.0), checkmate, tinytest

Encoding UTF-8

RoxygenNote 7.2.1

VignetteBuilder knitr

Collate 'bReplace.R' 'con2fcoales.R' 'valReplacement.R' 'whichDups.R'

Config/testthat/edition 3

NeedsCompilation no

Author Bandur Dragos [aut, cre]

Maintainer Bandur Dragos <dbandur@sympatico.ca>

Repository CRAN

Date/Publication 2022-08-19 23:10:02 UTC

Contents

bReplace *Batch-file value replacement*

Description

User-intended function to process a list of pairs of data files and associated lookup files listed in this order.

Usage

```
bReplace(dir, x, save = TRUE, msgs = FALSE)
```

Arguments

dir	Quoted character of length = 1 describing the path to the directory containing the data and associated lookup files, with either forward or double backward slash and no end slash (e.g. *"C:/path/to/directory"*).
x	List of character vectors each of length 2 containing full names of the data file and the associated lookup file, as described in replaceVals.
save	Logical, default TRUE: save results to directory; FALSE: display only.
msgs	Logical, default FALSE: suppress messages. TRUE: print a named list containing messages specific to each run.

Value

A named list displaying updated data and multiple replacement count tables. Also, updated csv files which are saved to *dir*.

Note

In examples, please leave argument *save* to FALSE. Otherwise, copy all content of folder "extdata" found in the installed package root into a directory on your machine. Use the absolute path to this directory as *dir* argument.

See Also

replaceVals

Examples

```
if (interactive()) {
# A list of data/lookup names:
fs = list(c('data.csv', 'lookup.csv')
        , c("data_unique.csv", "lookup_unique.csv")
        , c('data_id.csv', 'lookupNA.csv')
        , c('data_id.csv', 'lookupDUP.csv')
        , c('chile.csv', 'chile_nadup.csv')
```

```
        , c('data_id.csv', 'lookup_id.csv')
        , c('data_id.csv', 'lookup_idsimple.csv')
        , c('chile.csv', 'chile_id.csv')

    )
##Not run:
dir = system.file("extdata", package = "replacer")
bReplace(dir, fs, save = FALSE, msgs = TRUE)
}
```

`con2fcoales` *Helper for coalescing vectors of different types*

Description

This helper prevents the error in fcoalesce when attempting to coalesce two vectors of different data type (double/integer).

Usage

```
con2fcoales(u, z)
```

Arguments

u, z	Vectors of equal length and of different data types (e.g. double and integer). Missing values are accepted.

Value

A double data type vector of same length as the arguments.

`replaceVals` *User-intended wrapper for single-file replacements*

Description

The function sends the prepared data.tables to sReplace, receives updated data, displays a list of updated data and of counts of multiple replacements and saves updated data to disk (see Details).

Usage

```
replaceVals(dir, ..., save = TRUE)
```

Arguments

`dir`	Quoted character of length = 1 describing the path to the directory containing the data and associated lookup files, with either forward or double backward slash and no end slash (e.g. *"C:/path/to/directory"*).
`...`	Not used when file names are "data.csv", "lookup.csv". Otherwise, custom names including file extension, within quotation marks, such as "<data_name>.csv", "<lookup_name>.csv", **entered in this order!**.
`save`	Logical, default TRUE: save results to *dir*. FALSE: display only. See Note below.

Details

The workflow:

Tasks:

The function reads the data/lookup pair converting each file to "data.table" class, performs conformance checks on associated lookup, removes uninvolved data columns and non-standard lookup columns. Upon return from sReplace, re-structures updated result in the original format, saves the updated data to *dir* and displays a one-run named list containing updated data along with counts of duplicated and/or missing values replacements requests.

Messages:

The function displays messages and comments regarding the internal workflow. It is recommended reading these messages/comments as first troubleshooting step since they are specific to each file pair and request type. To suppress messages, wrap the function with suppressMessages. The vignette contains definitions of terms.

Value

A named list containing updated data and multiple replacement counts. Also, a csv file saved in the same directory, under the name updated_<data_name>*using*<lookup_name>.csv.

Note

In examples, please leave argument *save* to FALSE. Otherwise, copy all content of folder "extdata" found in the installed package root into a directory on your machine. Use the absolute path to this directory as *dir* argument.

See Also

bReplace, sReplace

Examples

```
## Not run: datasets with default names "data.csv", "lookup.csv" located in *dir*

if (interactive()) {
 dir = system.file("extdata", package = "replacer")
 replaceVals(dir, save = FALSE)
```

```
 ## no messages (not recommended!)
suppressMessages(replaceVals(dir, save = FALSE))
}
```

`sReplace` *Helper function for value replacement*

Description

The function is not intended for direct use. Once called by replaceVals it firstly checks for index presence in lookup. Upon the result of this check, the function moves along the branches of a decision tree (see Details).

Usage

```
sReplace(x, y0, uv)
```

Arguments

`x`, `y0`	Data.tables
`uv`	Character vector or list of same length as x, containing unique names of involved columns in data.

Details

The function starts by checking the presence of a User-made index in lookup.

If the index is found absent:

The function calls the helper whichDups to find the duplicated values in data. Also, looks for missing values set for *multiple* replacements and for eventual splits on missing data. In case of mixed *simple/multiple* requests the function splits lookup into maximum 3 subsets: one for *simple* replacements, for which it creates an internal index, one for *multiple replacements* of duplicated values for which it creates an internal index, and one for *multiple replacements* of missing values for which an internal index is not necessary.

Index for multiple replacements of duplicated values:
The internal index contains row numbers corresponding to all the elements of distinct subsets of duplicated values found within each involved data column and loops the function `data.table::set()` to perform replacements on these columns.

No Index for multiple replacements of missing values:
As mentioned above, no index is created for multiple replacements of missing values as there is only one generic value per data column. The missing values data subset is then *reshaped*, and the columns are *coalesced* (see `data.table` Manual) with corresponding data columns, for each generic value entered in lookup.

Index For Unique Values:
As stated above, simple replacements of unique values without User-made index are possible. Once the internal index created, the subset is *reshaped*, *joined* with the data on index and the corresponding columns are *coalesced*.

If the index is found present:

The function subsets the lookup using the special index values **0** and/or **NA** (or empty). At maximum, 3 subsets of lookup are formed as above. The replacement process is similar with the process used for absent index with the difference that simple replacements already have User-made index.

Value replacement:

Following the decision tree described above, the function calls utility's helpers and functions imported from the **data.table** package to process all lookup requests, in one single run.

Value

A named list containing updated involved columns in x, count of multiple replacements of duplicated values (if requested), count of multiple replacements of missing values (if requested).

See Also

dcast, fcoalesce, merge, set

`whichDups` *Find duplicated values in data*

Description

The function finds duplicated values in each column of the data file. Although not intended for direct use, it can be applied to a data file once converted into "data.table" class.

Usage

```
whichDups(x)
```

Arguments

`x`	A data.table.

Value

A named character vector. Data columns containing distinct sets of duplicated values have the names indexed.

Examples

```
if (interactive()) {

  dir = system.file('extdata', package = 'replacer')
  setwd(dir)

  x = data.table::fread('data.csv', na.strings = c(NA_character_, ''))

  whichDups(x)

}
```

Index